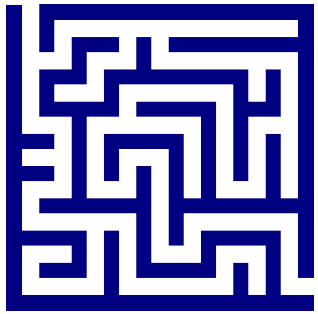
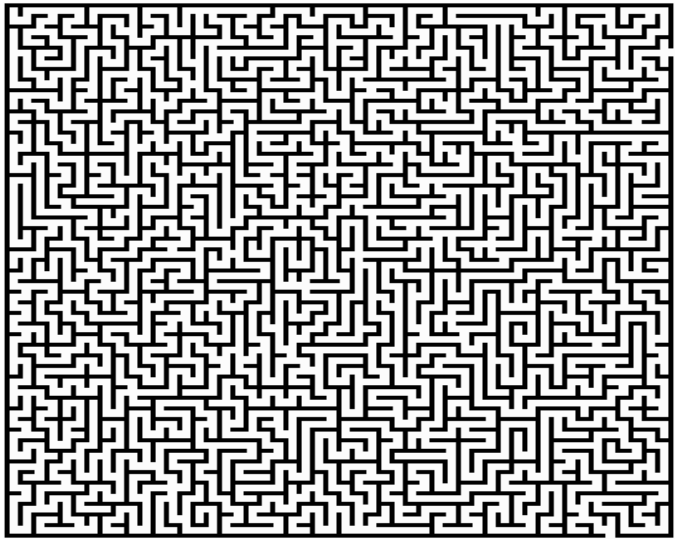
The Amaze Language

Project Manager	Rouault Francoeur
Language Guru	Daniel Mercado
System Architect	Jonathan Bourdett
System Integrator	Orlando Pineda
Tester and Validator	Jose Contreras

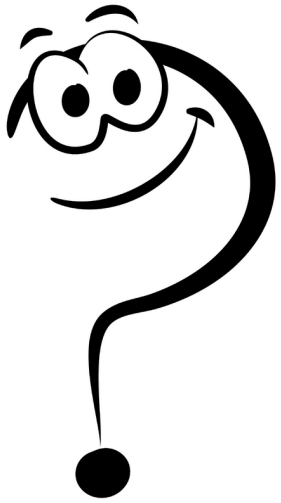


Motivation

- Interested in mazes and designing them.
- Wanted a way to design mazes for GUIs without the complexity of Java classes.
- Wanted to design an easy to learn, but hard to master language.



What is Amaze?



- Simple
- Intuitive
- Educational
- Architecture Neutral
- Detailed
- Creative
- Imperative and Domain Specific Programming Language used for designing mazes

Project Management

- Each member assigned responsibilities according to their role
- Members allowed flexibility to get work done efficiently
- Set long term goals and short term goals

Syntactic Constructs

1. Points
2. Paths
3. Structures
4. Functions
5. Boards
6. Conditionals / Iteration
7. Main Declaration

1. Points

point x : 1,1;

- Stores a two-dimensional location on the board. Location must be an integer

2. Path

path y : x, down, 1;

- Creates a line or "path" in the maze from a start location in a specified direction for a specified length.

- Directions accepted are "up", "down", "left", "right".

3. Structures

```
structure  line1 {
    point x : 1, 1;
    path y : x, right, 10;
}
```

- Reusable blocks of code that can set paths in multiple boards to avoid redundancy

4. Functions

```
func int foo (int x) {
    x = x + 7;
return x; }
```

- Takes an integer, increments it by 7 and returns it to the function call.

```
print "Hello World";
```

- Prints to the command line "Hello World";

5. Board

```
board stage1 {
    point x: 1, foo(8);
    path y: x, right, 10;
    set(line1);
}
```

- Create a frame where the maze is drawn.
- Accepts points and path declaration.
- Structures can be added using set().
- Functions can also be called in a board

6. Conditionals/ Iteration

```
int x = 1;
while( x < 3) {
    if (x == 1) { print "One"'; }
    else {print "Not One"; }
}
```

- if/else and while statements are made like in C except brackets are mandatory.
- else if statements do not exist in our language.

7. Main Declaration

```
main {
    draw(stage1);
}
```

- Although boards can be defined they must be drawn in the main function.
- Multiple boards can be drawn at the same time.

Sample Program

```
func int frank(int x){
      x = x + 7;
      return x;
}
structure megaman{
    point e: 0,0;
    point f:50,0;

    path g: e,right,50;
    path h: f,down,50;
```

Sample Program (Continued)

```
int bob = 0;
   int joe = frank(bob);

     if(true){
      while(bob < joe) {
```

Sample Program (Continued)

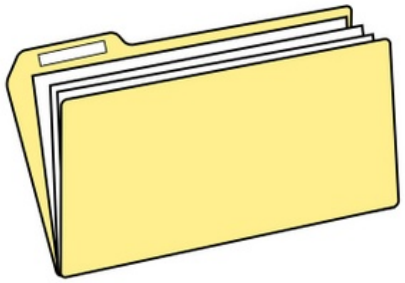
```
if(bob == 3){
                print("Swagneto");
            }
            else{
            point fred: bob, 9;
            path batman: fred,down,13;
        }
        bob = bob + 1;
    }
  }
}
```

Sample Program (Continued)

```
board d {
    size: 100,100;
    start:0,0;
    end:50,50;

     set(megaman);

}

main {
     draw (d);
}
```

Compiling and Running Amaze

- /src folder contains the source files of the Amaze compiler
- run amazec.sh with an .amz file as an input
- ./amazec.sh input_file.amz
- a Java executable output to the src folder
- execute with Java
- java Out



Sample Programs

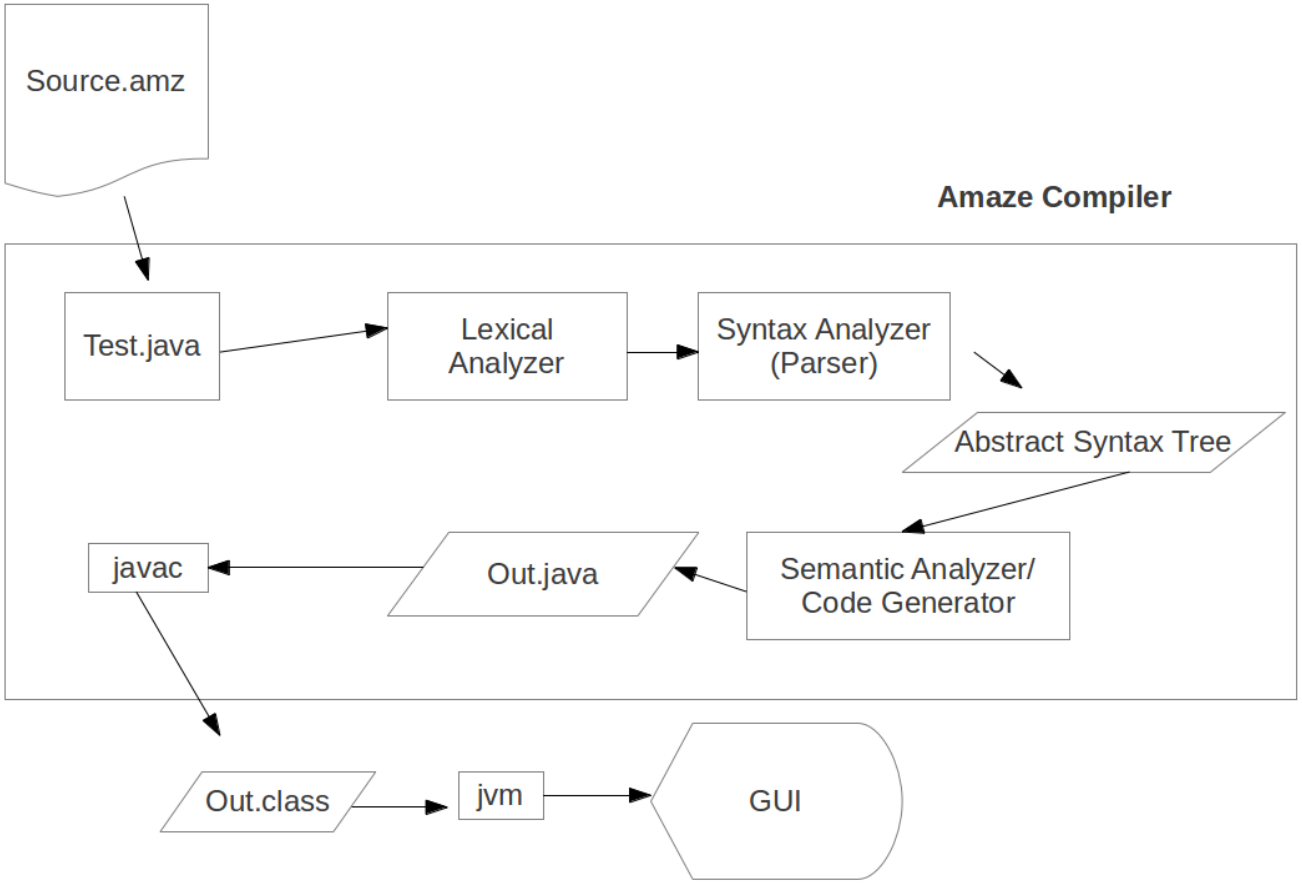
Demo1

Maze showing using a structure type

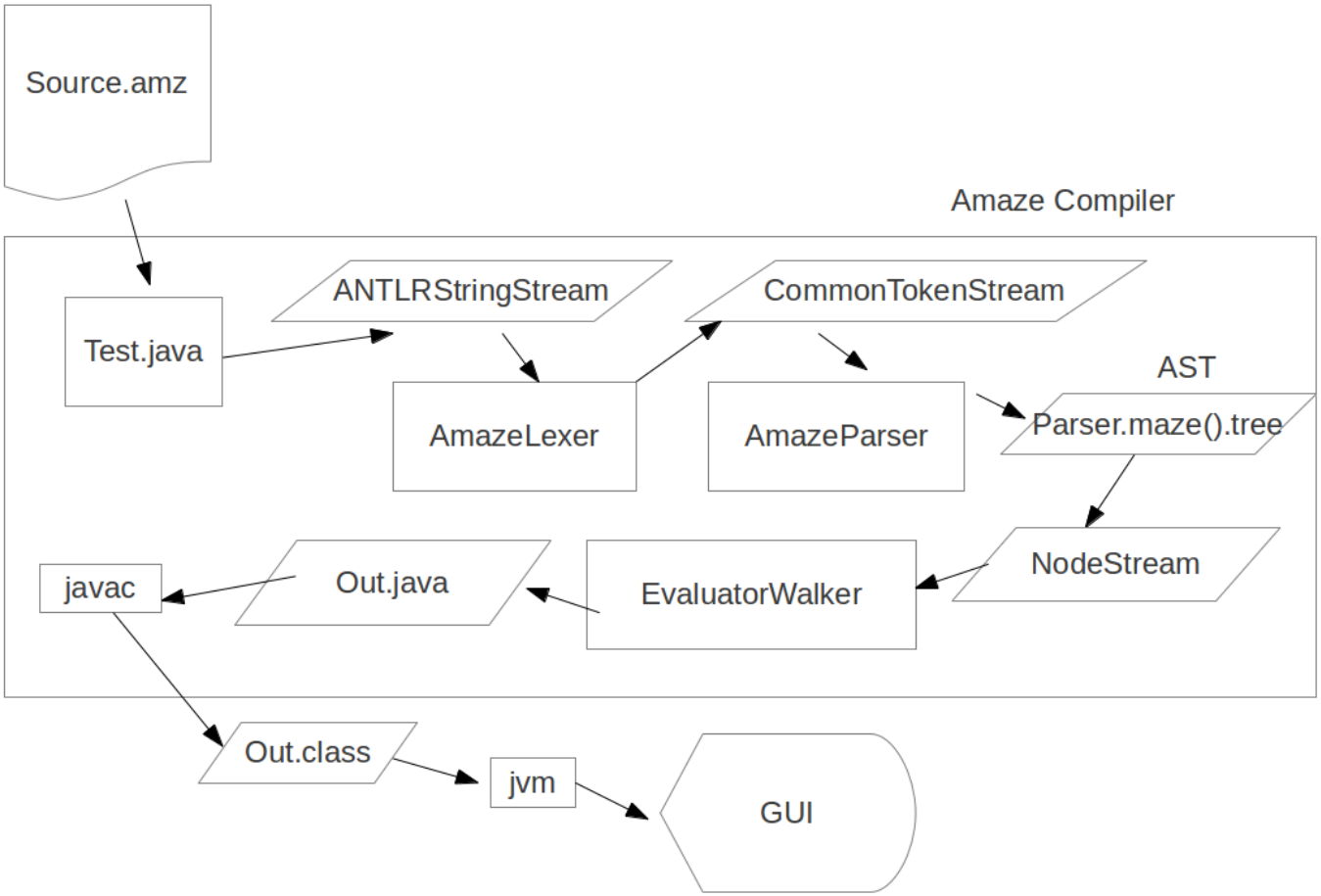
Demo 2

"Maze" showing a print statement, function, iteration, conditional statement, and structure type

Translator Architecture



Translator Architecture (More Specific)



Software Development Environment

Atlassian
bitbucket

ANTLRWorks
Edit, visualize and debug ANTLR grammars
Version 1.4.3
(c) 2005-2011 Jean Bovet & Terence Parr

Test Plan

white-box testing

```
board board1
{
        size: 19, 19;
        start: 1, 0;
        end: 18, 17;
        int x = 0;
        while(x)
        {
            x=10;
        }

}
main
{
        draw(board1);
}
```

amazec

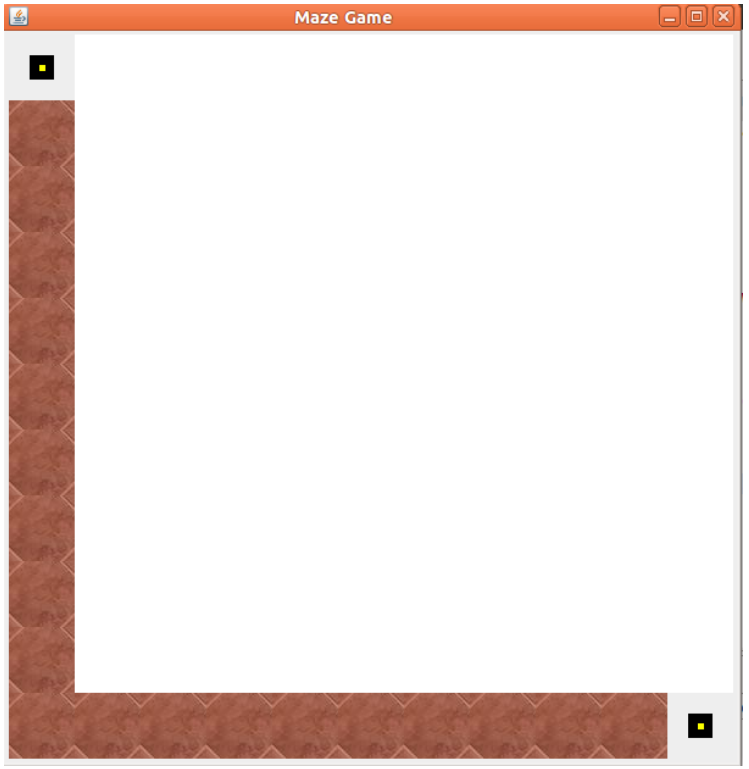
"Invalid type in while loop. Expected a boolean but received a(n) int"

Test Plan

black-box testing

```
board board1{
        size : 11, 11;
        start : 0,0;
        end : 10, 10;
        point p : 0,0;
        path myPath : p, down,
10;
        point p2: 0,10;
path myPath2:p2,right,10;

}

main{
        draw(board1);
}
```

amazec



Test Plan

Unit Testing

Regression Testing

Lessons Learned

- Start Early
- Set Milestones
- Communicate Effectively
- Figure Out How to Work in Parallel
- Learn the capabilities of your team mates
- Communicate with other peers
- Don’t be afraid to use new tools or reject old tools!

What Worked Well

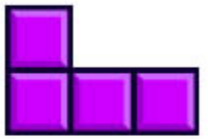
- Meeting Every Week
- Team Synergy
- Comfortable Atmosphere
- Learning the Tools
  - ANTLR
  - git

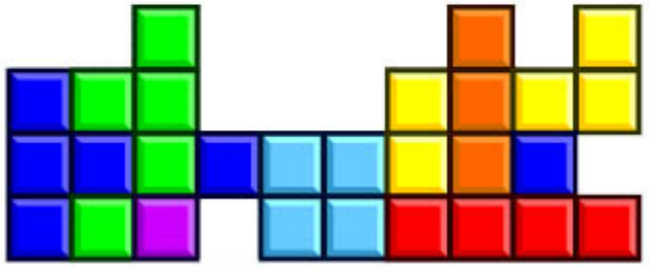
Why use Amaze language?

- Fun
- Simple
- Educational
- Exercise your Creativity
- Intuitive
- Easy to Demonstrate

Hopes for the future

- Expand idea to a language made to design interactive maze games
- game design language





THE END

Questions?